Optimizing Triangulations By Curvature Equalization

Lori L. Scarlatos
Grumman Data Systems
1000 Woodbury Rd., D12-237
Woodbury, NY 11797

Theo Pavlidis
State University of New York
Department of Computer Science
Stony Brook, NY 11794-4400

Abstract

Triangulated irregular networks (TINs) are an attractive form of surface approximation because triangle vertices and edges may be adaptively selected to produce a good fit with a minimal number of triangles. Finding methods for selecting these vertices and edges, however, is still an active area of research. Numerous refinement algorithms have been proposed, but these may produce more triangles than necessary. In this paper we present an algorithm that attempts to improve a triangulation by shifting the vertices so that curvature within the triangles is nearly equal. In addition, unnecessary triangles are removed. We finish with results produced by running this algorithm on simple geometric surfaces and real terrain data.

1. Introduction

Triangulated irregular networks (TINs) provide an excellent means of approximating surfaces. Any three points on a surface may provide the three vertices of a triangle without restrictions on placement or size. This gives us the potential to produce a good approximation while minimizing the number of surface patches required. However the optimal placement of triangle vertices and edges in a triangulation is still an area of active research.

Many triangulation techniques rely on refinement methods to find a set of triangles that approximates a surface to within given error constraints [1-5]. In general, these methods start with a coarse triangulation and iteratively refine it by strategically adding points used to create new triangles in the model. One of these methods was developed by the authors [5].

Although adding points in such a manner does eventually produce the desired goodness-of-fit, this does not guarantee that the resulting set of triangles cannot be improved. For example, the triangulation may contain more triangles than are needed, which would increase storage space and time required to render the surface. It may also contain too many slivery triangles - characterized by at least one very acute angle - which cause artifacts in the display and anomalies in some analysis functions like finite element analysis.

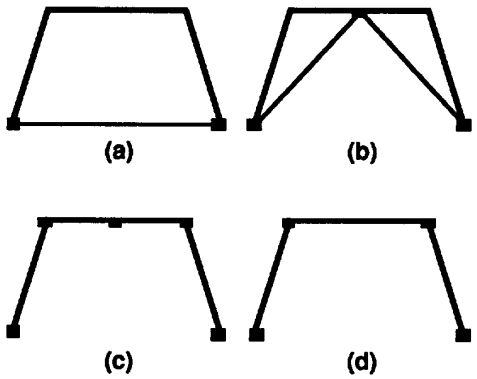

Figure 1. Approximating a curve using split-and-merge refinement.

An analogy to this problem may be found in the one-dimensional case where a curve is approximated with a series of straight line segments, as shown in Figure 1. Starting with a line segment connecting the two endpoints of the curve (Figure 1a), an approximation may be produced by successively splitting line segments at points of greatest error (Figure 1b) until the line segments fit the curve within a given tolerance (Figure 1c). Although the resulting approximation is error-free, it contains more points than necessary. In the one-dimensional case, this problem is solved by merging line segments (Figure 1d). As shown, the two middle segments are merged so that the curvature of each interval represented by a line segment is roughly the same. Pavlidis' text [6] describes this split-and-merge technique in greater detail.

The literature on polygonal approximations is extensive [6, 10-14]. Some of these methods are based on the result that in an optimal polygonal approximation vertices are placed so that an integral of the curvature takes the same value over all intervals [10]. In contrast, the literature on triangulated approximations is comparatively limited [7-9]. Extending polygonal approximation techniques to triangulations is difficult because there is no direct counterpart of merging triangles when we look at the two-dimensional case. Consider, for

0-8186-2897-9/92 $03.00 © 1992 IEEE

example, the solid in Figure 2 which has a square base and a smaller square top. An initial triangulation may be created by connecting the four base points to a point of greatest error found anywhere on the top plane. As shown, using such an initial approximation can produce more triangles than necessary.

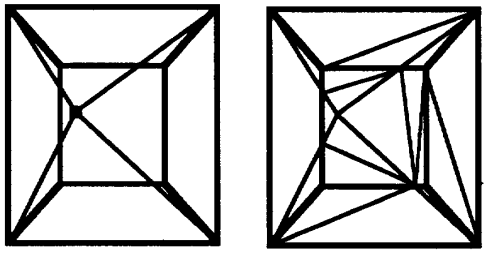

Figure 2. Surface approximations using refinement techniques can produce more triangles than necessary.

We have developed a novel approach to triangulation optimization that extends the ideas of polygonal approximation. Since it is not feasible to extend the split-and-merge algorithm to triangulations, we follow the alternative strategy of moving vertices. We move these vertices - and collapse very thin pairs of triangles - so that the triangles approximate surface patches of similar curvature.

This paper describes our strategy for moving triangle vertices to produce the desired approximation with a minimal number of triangles. Although our strategy is to equalize curvature, our goal is to produce a triangulation that meets given error constraints using as few triangles as possible, all as nearly equilateral as possible. We assume that the triangulation corresponds to a known underlying surface or bivariate function which is sampled at regular intervals, such as a digital terrain model. This essentially extends the work of McClure and Shwartz [7], presenting an algorithm for producing triangulations that meet their criteria for good surface approximations.

In the next section, we briefly describe the theoretical work of McClure and Shwartz [7] which forms the foundation of this work. The section following that presents our practical approach and implementation of that concept. Results of the implementation, applied to both geometric test cases and real-world terrain data, are described in the last section.

2. Foundations

In a recent paper [7], McClure and Shwartz discuss methods of surface reconstruction based on triangulations. Although their focus is on image data compression, they view this as being analogous to the problem of defining concise and accurate approximations for surfaces. One of the questions that they explore is how fine a regular triangulation needs to be in order to provide a consistent representation of an underlying surface. To determine this level of detail, they develop an estimator for the surface which is based on the Hessian matrix of second partial derivatives $H(p_{i,j})$. On a discretely sampled surface where each $p_{i,j}$ maps to a single elevation $f(x_i,y_j)$, the second partial derivatives of $p_{i,j}$ may be approximated as follows:

$$f_{xx} = f(x_{i-1},y_j) + f(x_{i+1},y_j) - 2f(x_i,y_j)$$

$$f_{yy} = f(x_i,y_{j-1}) + f(x_i,y_{j+1}) - 2f(x_i,y_j)$$

$$f_{xy} = f_{yx} = f(x_{i-1},y_{j-1}) + f(x_{i+1},y_{j+1}) - f(x_{i-1},y_{j+1}) - f(x_{i+1},y_{j-1}).$$

Then for each point $p_{i,j}$, the curvature estimator at that point may be expressed by

$$MS(p_{i,j}) = 3(\text{trace } H(p_{i,j}))^2 - 8(\det H(p_{i,j})).$$

Although this is not curvature in the classical sense, we use the term loosely because this measure detects the salient features of a surface. In fact, the above measure incorporates both the mean curvature (approximated by the trace of the Hessian) and the Gaussian curvature (approximated by the determinant of the Hessian). This improves on simpler measures such as the determinant of the Hessian which fail to detect many critical features in real-world situations. Consider, for example, the edge of a cliff as shown in Figure 3. Although points along this edge are clearly critical to the model, the determinant of the Hessian at these points is zero.

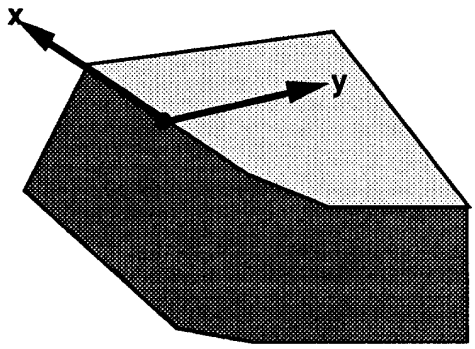

Figure 3. The Hessian of a point on the edge of a cliff may be zero.

Given the curvature of each point on the surface, the curvature of a triangle can be expressed as an integral of this measure over the entire triangle. When the surface is represented by a set of discrete sample points, the

curvature of each triangle may be approximated by summing these measures for all points within the triangles.

In their extensions section of [7], McClure and Shwartz discuss how this estimator relates to the selection of nonhomogeneous (irregular) triangles for approximating the surface. Although they provide an expression for the ideal density of distributed sample points and triangles, they do not give an actual algorithm for selecting these points and triangles. This paper extends their work by providing one such algorithm.

3. Approach

Refinement techniques often fail to find the optimal solution because critical features are not always evident at the coarser levels of detail. Instead, several refinement iterations are required before these features are revealed.

Our approach is to start with a triangulation that already meets error constraints, but is not optimal. It may, for example, contain more triangles than are needed and/or too many slivery triangles. We assume that this triangulation approximates a surface represented by regularly spaced discrete sample points, such as those found in a digital terrain model. Error in the triangulation is measured by projecting each of these sample points to the appropriate triangle.

We attempt to improve the model by moving triangle vertices so that curvature within each of the triangles is as nearly equivalent as possible without introducing errors to the model. The curvature of each triangle is approximated by summing the curvature estimates for all sample points interior to the triangle. Points coincident with triangle edges are not included in these sums. Indeed the goal of the algorithm is to place such boundaries over areas of high curvature, keeping triangle interiors relatively flat.

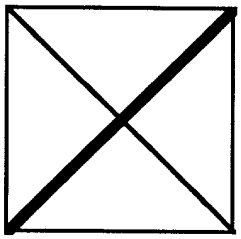

Figure 4. A triangulation with equalized curvature may contain too many triangles.

We use a two-step procedure for moving vertices. The algorithm of the first step shifts triangle vertices, attempting to equalize the curvatures within the triangles. However, as shown in Figure 4, the resulting triangulation may still contain too many triangles. In this picture, points of high curvature occur only along the ridge (represented by a dark line). This surface would be best represented with two triangles with a shared edge corresponding to the ridge. Our algorithm for the second step remedies this by attempting to remove pairs of triangles without introducing additional errors to the model. Both algorithms are described in greater detail below.

3.1. Equalizing Curvature

The algorithm of the first step attempts to equalize curvature by iteratively reducing the size of the triangles with greatest curvature. This size reduction is achieved by moving each vertex of the triangle inward, one at a time. If a neighbor sharing that vertex also has high curvature, then the point is moved along the common edge. Otherwise, it is moved along the bisectrix of that angle of the triangle. Note that points along the boundary of the domain of the triangulation may only move along that boundary, and that points defining the corners of the boundary may not move at all.

Initially the point is moved by some set distance. If this move makes the triangle very thin and slivery then we collapse the triangle and one of its neighbors, effectively removing them both. This is done by merging two of its vertices as shown in Figure 5.

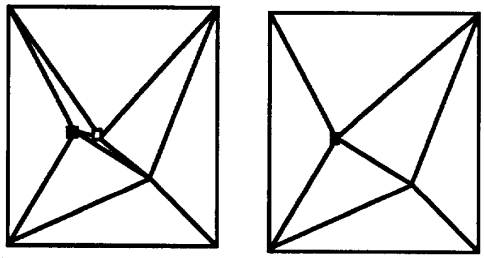

Figure 5. Slivery triangles are eliminated by sliding one vertex along an edge and merging it with another vertex .

After determining where the vertex moves to, we recalculate the curvatures of the affected triangles to see if the overall curvature of the surface is more equalized. Because points on the triangle edges do not contribute to this overall curvature measure, positioning triangle edges over critical edges on the surface will reduce overall curvature for the surface. If the resulting triangulation is no worse than the previous one, then the triangulation is updated to reflect this move. Otherwise, we try moving the vertex half the distance we tried earlier. Eventually, either the vertex will move, or the distance will become

negligible, causing the algorithm to skip to the next triangle vertex.

Because we allow moves that make the triangulation no worse, there is a danger of cycling. To prevent this, we keep a record of all attempted moves. Any move that has been tried before is automatically rejected, and the algorithm proceeds to the next triangle/vertex.

Sometimes we are unable to move any of the vertices on the triangle with greatest curvature. This is generally due to constraints imposed by the area border. Consider, for example, the ridge in Figure 6, indicated by the dark line. Because points of high curvature occur only along this ridge, curvature cannot be equalized in any triangulation in which this ridge corresponds to a triangle edge. Hence, this algorithm can only go so far in equalizing the curvatures.

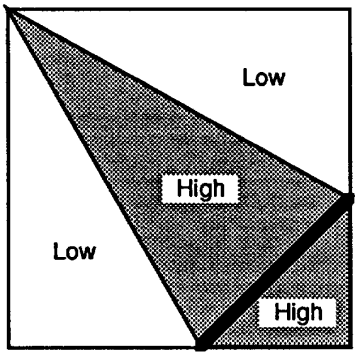

Figure 6. Due to constraints imposed by a finite boundary, curvature cannot be equalized without introducing errors.

Our algorithm continues to try moving vertices until the triangle with "maximum" curvature has curvature equal to the smallest curvature in the triangulation. If it is unable to move any vertices on the triangle of highest curvature, it proceeds to the triangle of next-highest curvature and considers that as the triangle of "maximum" curvature. Because the algorithm skips over triangles in this manner - and doesn't allow repeated moves - either the curvature within the triangulation will become equal, or the "maximum" triangle will eventually be the same as the "minimum", and the algorithm will halt.

This algorithm for the first step is summarized in the pseudo-code below.

```
EQUALIZE_CURVATURES(Point, Npt, Triangle, Ntri,
Member, Curvature, TriList)
/* Point - array of Npt points (x,y,z) */
/* Triangle - array of Ntri triangles, each with 3 point & 3
neighbor references */
/* Member - triangle(s) each point projects to */
/* Curvature - overall curvature of each triangle */
/* TriList - triangles using each point as a vertex */
{
  Sort triangles on Curvatures in descending order;
  max_tri = triangle at head of the sorted list;
  min_tri = triangle at tail of the sorted list;
  While Curvature[max_tri] > Curvature [min_tri]{
    For each point P on Triangle[max_tri] {
      Set distance;
      Try moving point P inward by distance to point Q;
      If P can't be moved or this was already tried then {
        If no vertices on this triangle could move then
          max_tri = next triangle on sorted list;
      }
      Else repeat {
        Record attempt to move P to Q in history;
        Determine whether resulting new triangle is too
        skinny and must be removed;
        If this is a good move to make then {
          Update triangulation;
          Fix sorted list;
          Reassign max_tri;
        }
        else {
          Cut distance in half;
          Try moving P inward by distance to point Q;
          If P can't be moved or this was tried then {
            If no triangle vertices could be moved then
              max_tri = next triangle on list;
            Done with point P;
          }
        }
      } until vertex moves or Done with point P;
    }
  }
}
```

3.2. Removing Unnecessary Triangles

The algorithm of the second step attempts to further improve the triangulation by removing unnecessary pairs of triangles. Although this may increase the curvature measures within triangles, it must not increase actual error in the model. Error is measured by projecting points from the original discrete sampling to the triangulated surface and finding the difference.

Our algorithm removes two triangles by collapsing their common edge into a single point as shown in Figure 5. It tries to remove every pair of triangles within the triangulation, resulting in less than 3/2T iterations, where T is the number of triangles initially. Because an edge may be collapsed into either of its endpoints, the

algorithm tries merging both ways and chooses the better way.

This algorithm for the second step is summarized in the pseudo-code below.

```
REMOVE_UNNECESSARY_TRIANGLES(Point, Npt,
Triangle, Ntri, Member, Curvature, TriList)
/* Point - array of Npt points (x,y,z) */
/* Triangle - array of Ntri triangles, each having 3 point & 3
neighbor references */
/* Member - triangle(s) each point projects to */
/* Curvature - overall curvature for each triangle */
/* TriList - triangles using each point as a vertex */
{
  For each edge PQ shared by 2 triangles {
    Try moving point P to point Q;
    Try moving point Q to point P;
    Select the better move to make;
    If this doesn't introduce errors then {
      Update triangulation;
    }
  }
}
```

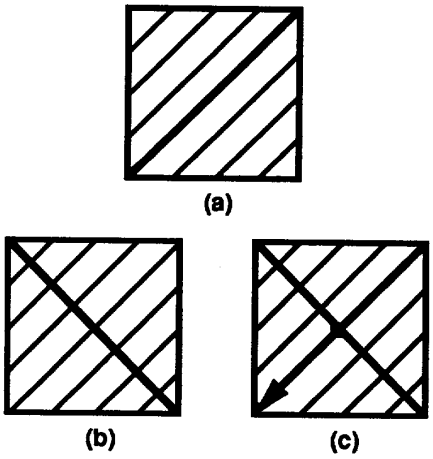

Figure 7. Limitations.

3.3. Limitations

There are limitations on this approach imposed by the constraint that points must never leave the edge bounding the convex hull of the triangulation. Consider, for example, the surface triangulation depicted in Figure 7a, where a ridge runs along the diagonal. If the initial triangulation has a single edge crossing the ridge perpendicularly as shown in Figure 7b, then none of the vertices may be moved. The only solution is to use a triangulation that has considered the surface features, such as [5]. Then the situation shown in Figure 7c would occur, and the center vertex could slide along the ridge to produce the desired triangulation.

4. Results

We first ran our algorithms using the test samples illustrated in Figure 8. These samples represent a ridge, a pyramid, and a paraboloid. In this figure, column A shows the original triangulations. Column B shows the results after equalizing curvature in the first step of our procedure. Column C shows the results after removing unnecessary triangles in the second step of our procedure. In these diagrams darkened lines represent ridges and highlighted points move in the next step. As shown, our algorithms produced optimal triangulations in all cases. This demonstrates that our methods work well for curved as well as polyhedral surfaces.

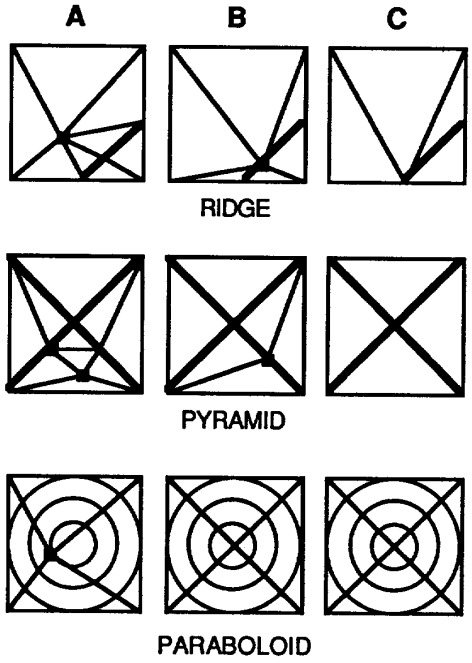

Figure 8. Results of applying the algorithm to artificial surfaces.

Next we ran our algorithms on triangulations produced by our program [5] which considers terrain features in the triangulation. Although this algorithm produces good triangulations within a given error tolerance, we suspected that by moving the vertices we could reduce sliveriness of the triangles as well as the number of triangles in the model.

We ran our program on 24 triangulations representing different types of terrain and different degrees of refinement. We then measured the success of this operation using 3 criteria: number of triangles removed, reduction in sliveriness, and reduction in error of the model.

In our tests, our algorithm reduced the number of triangles in 70% of the triangulations tested. In the best case, the final triangulation contained 22% fewer triangles. However in most cases the reduction was nominal. As expected, the greatest reductions occurred with triangulations of terrain with sharp features such as plateaus and mountain ridges, and on coarser triangulation models.

Slivery triangles - characterized by one very acute angle - produce visual artifacts in the rendered surface model and anomalies in some analysis functions such as finite element analysis. It is therefore desirable to reduce sliveriness. We measure sliveriness of a triangle with the following value, which is smallest when the triangle is equilateral:

$$\frac{\text{Perimeter}^2}{\text{Area}}$$

However, as shown in [9], slivery triangles are sometimes inevitable in an accurate surface model. In our tests, we found that the triangles became less slivery in 50% of the cases. The greatest improvement occurred for a region of steep plateaus, where average sliveriness was reduced by 50% and maximum sliveriness was reduced by 77%.

While using this real data, we discovered that equalizing curvature may indiscriminately add intolerable errors to the surface model. However, we found that by testing for this possibility, we are able to equalize curvature without adding error to the model. In fact, the statistics on the final curvature (maximum, average, and variance) were the same as for when the errors increased. Maximum error in the model actually decreased in 20% of our test cases.

Figure 9 illustrates results in one case where this algorithm reduced sliveriness and error. Figure 9a shows the original data for a region in Nevada. This triangulated regular grid contains 10,952 triangles. Figure 9b shows a triangulation of this surface produced by the algorithm in [5]. This triangulation contains 1,998 triangles, a compression ratio of approximately 11:2. Maximum and average error in this model are 19 and 1.98 meters respectively. Figure 9c shows this triangulation after moving the vertices. With the same number of triangles, maximum and average error are reduced to 15 and 1.89 meters respectively. Additionally, average sliveriness is reduced by 20%.

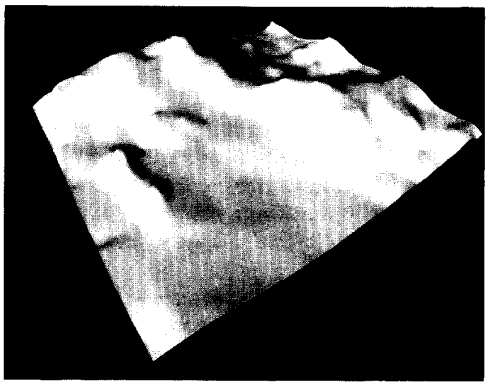

Figure 9a. Original Data

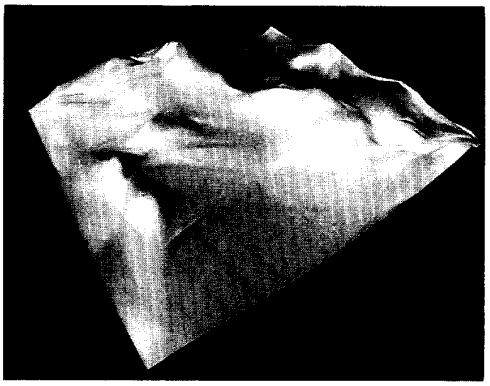

Figure 9b. Triangulation Before Moving Vertices

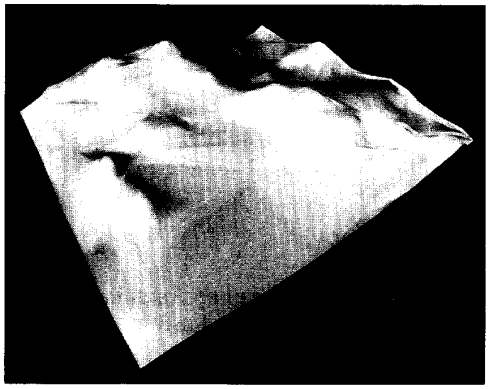

Figure 9c. Triangulation After Moving Vertices

5. Conclusions

Our method for moving triangle vertices is an effective way of guaranteeing that the triangle vertices are points of highest curvature and triangle edges correspond to distinctive edges on the surface. Triangulations of surfaces with constant curvature - and hence no distinctive features - will gain nothing from this algorithm, or any other optimization algorithm for that matter.

As demonstrated by our results, our technique of moving triangle vertices can improve some triangulation models. Greatest improvements occur with surfaces characterized by sharp edges, such as the pyramid and ridge models. Less improvement occurs on models that already approximate the surface topology and/or have less distinctive features, such as our terrain model examples. We believe that this method could be effectively used to create an initial triangulation for our hierarchical triangulation technique [5] or to improve a regular triangulation of a surface. Because this technique works best with surfaces with sharp features, we expect that this technique could be used to improve three-dimensional surface models for CAD, animation, and three-dimensional scientific visualization.

6. Acknowledgements

The authors would like to acknowledge the help of Professor D.E. McClure of Brown University for providing us a preprint of [7] and also pointing out additional references in the appropriate literature. We also want to thank R. Kelly and H. Tesser of Grumman Data Systems for their continued support.

7. References

[1] Fowler, R.J. and Little, J.J, "Automatic Extraction of Irregular Network Digital Terrain Models", *Proceedings of SIGGRAPH '79*, pp. 199-207, 1979.

[2] DeFloriani, L., Falcidieno, B., Nagy, C., and Pienovi, C., "A Hierarchical Structure for Surface Approximation", *Computers and Graphics*, 8(2), pp. 183-193, 1984.

[3] DeFloriani, L., "A Pyramidal Data Structure for Triangle-based Surface Description", *IEEE Computer Graphics & Applications*, 9(2), pp. 67-78, 1989.

[4] Scarlatos, L.L., "A Refined Triangulation Hierarchy for Multiple Levels of Terrain Detail", *Proceedings of IMAGE V*, pp. 115-122, 1990.

[5] Scarlatos, L. and Pavlidis, T., "Hierarchical Triangulation Using Cartographic Coherence", *CVGIP: Graphical Models and Image Processing*, 54(2), pp. 147-161, 1992.

[6] Pavlidis, T., *Structural Pattern Recognition*, Springer-Verlag, New York, 1977.

[7] McClure, D.E. and Shwartz, S.C., "A Method of Image Representation Based on Bivariate Splines", pre-print, 1988.

[8] Nadler, E., "Piecewise Linear Best L2 Approximation on Triangulations", in *Approximation Theory V*, edited by Chui, C.K., Schumaker, L.L., and Ward, J.D., Academic Press, New York, 1986.

[9] Dyn, N., Levin, D., and Rippa, S., "Data Dependent Triangulations for Piecewise Linear Interpolation", *IMA Journal of Numerical Analysis*, 10, pp. 137-154, 1990.

[10] McClure, D.E., "Nonlinear Segmented Function Approximation and Analysis of Line Patterns", *Quarterly of Applied Mathematics*, 33, pp. 1-37, 1975.

[11] Slansky, J. and Kibler, D.F., "A Theory of Nonuniformly Digitized Binary Pictures", *IEEE Transactions on Systems, Man, and Cybernetics*, 6(9), pp. 637-647, 1976.

[12] Montanari, U., "A Note on Minimal Length Polygonal Approximation to a Digitized Contour", *Communications of the ACM*, 13, pp. 41-47, 1970.

[13] Tomek, I., "Piecewise Linear Approximations", *IEEE Transactions on Computers*, 23, pp. 445-448, 1974.

[14] Duda, R.O. and Hart, P.E., *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.